1-2007

An improvement heuristic for the timetabling problem

Aldy GUNAWAN
Singapore Management University, aldygunawan@smu.edu.sg

Kien Ming NG

Kim Leng POH

International Journal of Computational Science
1992-6669 (Print) 1992-6677 (Online) © Global Information Publisher
2007, Vol. 1, No. 2, 162-178

An Improvement Heuristic for the Timetabling Problem

Aldy Gunawan*, Kien Ming Ng, and Kim Leng Poh

Department of Industrial and Systems Engineering, Faculty of Engineering,
National University of Singapore, 1 Engineering Drive 2, S(117576), Singapore
{g0500710,isenkm,isepohkl}@nus.edu.sg

Abstract. This paper formulates a timetabling problem, which is often encountered in a university, as a mathematical programming model. The proposed model combines both teacher assignment and course scheduling problems simultaneously, which causes the entire model to become more complex. We propose an improvement heuristic algorithm to solve such a model. The proposed algorithm has been tested with several randomly generated datasets of sizes that are comparable to those occurring in a university in Indonesia. The computational results show that the improvement heuristic is not only able to obtain good solutions, but is also able to do so within reasonable computational time.

Keywords: Timetabling, mathematical programming, improvement heuristic algorithm.

1 Introduction

Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives [1]. One of the key applications of timetabling is in educational timetabling. In this paper, we focus solely on the course timetabling problem at the university level.

The course timetabling problem is further classified into five sub-problems: teacher assignment, class-teacher timetabling, course scheduling, student scheduling, and classroom assignment [2]. The teacher assignment problem focuses on allocating teachers to courses, without considering the allocation of courses to time periods. The course scheduling problem only focuses on allocating courses to time periods. It is often assumed that the allocation of teachers to courses has been done earlier before the actual scheduling of time periods to courses.

* Corresponding Author. Tel.: +65-6516-2208, Fax: +65-6516-1434, Email: g0500710@nus.edu.sg.

This paper attempts to address both teaching assignment and course scheduling problems simultaneously, and a model combining these two sub-problems is proposed. Another emphasis of this paper is to develop a simple improvement heuristic to tackle this timetabling problem. Such a timetabling problem that we address has arisen in the context of a university in Indonesia.

The paper proceeds as follows: Section 2 presents the timetabling problem and gives a brief literature review of this problem, as well as some of its basic requirements. Section 3 describes a mathematical programming model for the problem, together with the computational difficulties of the model. A proposed heuristic is then described in Section 4, and the computational results and comparisons are reported in Section 5. Finally, Section 6 gives some conclusions and possible directions for future research.

2 The Timetabling Problem

The course timetabling problem discussed in this paper appears in the context of a university in Indonesia. A number of common definitions and terms for the problem are first explained in detail. A course refers to a subject taught one or more times within a week. Each course requires a certain number of consecutive time periods per week. Due to the capacity of the classrooms and the number of students registered, some courses have to be taught repeatedly by the same teacher or by different teachers. Each of these repeated courses is known as a course section.

Before the new semester starts, teachers are requested to decide the courses they are willing to teach, along with their preferred days and time periods to teach the courses. The primary problem faced is how to assign teachers to their preferred courses and course sections and then to schedule course sections to time periods over a week based on the teachers' preferences. Although the decisions on teacher assignment and course scheduling problems could mathematically be made in one step, in practice, this is generally done in two separate steps [3]. Most of the papers only focus on one of the sub-problems, such as Andrew and Collins [3], Breslaw [4], Harwood and Lawless [5], Schniederjans and Kim [6], Tillett [7], and Wang [8] that only focus on the teacher assignment problem.

The course scheduling problem has also been widely studied with many solution methods being proposed. Yu and Sung [9] applied a genetic algorithm for solving course scheduling problems, and simulated annealing was applied to course scheduling by Abramson [10] and Abramson et al. [11]. Hertz [12] proposed a tabu search algorithm for solving course scheduling problems, while Badri [13] proposed a model for both course scheduling and teacher assignment problems. Through a two-stage optimization procedure, the model in Badri [13] seeks to maximize teacher-course preferences in assigning teachers to courses, and then maximize teacher-time preferences in allocating courses to time periods.

Two common techniques for solving the timetabling problems are exact and heuristic methods. Exact methods include formulating and solving integer programming models of the timetabling problems. One of the recent applications of integer programming to the course scheduling problem

was presented by Daskalaki et al. [14]. We adopt a similar approach by formulating a mathematical programming model that considers both teacher assignment and course scheduling simultaneously. This is also an extension of the basic mathematical programming model proposed by Gunawan et al. [15], in which it is shown that timetabling problems with data sizes comparable to that of an institution can be solved with the help of this basic model.

For large problems, it could be difficult to find and prove the existence of an optimal solution, especially within short computing times [16]. It would be necessary to develop a heuristic approach in order to find a good solution within a reasonable amount of time. In general, the heuristic procedure consists of two phases [17]: initial and improvement phases. In the first phase, we try to find a feasible solution. If there is no feasible solution, we need to rearrange the assignments or relax some requirements or constraints. In the improvement phase, we improve the feasible solution obtained until a local optimum is reached. Some examples of heuristic approaches for solving the course scheduling problem were proposed by Aubin and Ferland [18], Loo et al. [19] and Wright [20].

The university timetabling problem has special features that highly depend on the university's characteristics, such as the courses taught, the teachers and the availability of resources as well. Several rules or requirements imposed on the problem are as follows:

a. For each teacher, course and time preferences are obeyed.
b. For each course, only one section can be conducted in every time period.
c. Each teacher has to teach at least one course and cannot teach more than his/her maximum load. Here, the load refers to the number of courses that are assigned to the teacher.
d. The number of teachers who can teach for each course is limited.
e. All course sections have to be spread evenly throughout a week, so that for a particular course, only one section can be conducted every day.
f. Each teacher can only teach at most one course section in a particular time period.
g. The number of course sections taught cannot exceed the number of classrooms available during each time period.
h. All sections for a particular course must be scheduled.
i. Each course section can only be taught by one teacher.
j. Each teacher will not be assigned courses that he/she is unable to teach.
k. All the course sections taught by a teacher will be spread out evenly during a week.
l. Each course section has to be scheduled in a certain number of time periods consecutively.

The first requirement is considered as a preference and will be incorporated into the objective function, while the rest will be regarded as constraints that cannot be violated.

3 The Proposed Mathematical Programming Model

In order to study the computational effort involved in solving the problem of interest, the following mathematical programming model is proposed. We define the following sets to be used in the model:

I set of all teachers

J set of all courses

K set of all course sections

L set of all days available

M set of all time periods available

J_i set of courses that could be taught by teacher i ($i \in I$)

K_j set of sections of course j ($j \in J$)

We also define the following data parameters for the model:

N_i maximum load (in terms of the number of courses) of teacher i ($i \in I$)

C number of classrooms available per time period

H_j number of time periods required for course j ($j \in J$)

PC_{ij} value given by teacher i on the preference to be assigned to teach course j ($i \in I$, $j \in J$)

PT_{ilm} value given by teacher i on the preference to be assigned to teach in day l and time period m ($i \in I$, $l \in L$, $m \in M$)

LT_j minimum number of teachers that could teach course j ($j \in J$)

UT_j maximum number of teachers that could teach course j ($j \in J$)

S_j number of sections of course j ($j \in J$)

Finally, the following decision variables will be required to define the problem:

X_{ijklm}= 1 if teacher i teaches course j section k on day l and at time period m; 0 otherwise ($i \in I$, $j \in J$, $k \in K_j$, $l \in L$, $m \in M$)

Y_{ijkl}= 1 if teacher i teaches course j section k on day l; 0 otherwise ($i \in I$, $j \in J$, $k \in K_j$, $l \in L$)

U_{ijklm}= 1 if teacher i teaches course j section k on day l and started at time period m; 0 otherwise ($i \in I$, $j \in J$, $k \in K_j$, $l \in L$, $m \in M$)

P_{ij}= 1 if teacher i teaches course j; 0 otherwise ($i \in I$, $j \in J$)

L_i= number of course sections taught by teacher i ($i \in I$)

V_i= number of course sections taught by teacher i per day ($i \in I$)

For our problem, the objective function reflects a preference function that needs to be maximized. It refers to the total preferences of assigning courses to the teachers and the total preferences of assigning the days and time periods to the teachers for teaching the sections of the courses. Thus, the teachers can choose not only their course preferences, but they can also indicate their time preferences. We assume that these preferences are equally important. The objective function is described by the expression in equation (1):

$$\text{Maximize} \quad \sum_{i \in I} \sum_{j \in J} PC_{ij} \times P_{ij} + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K_j} \sum_{l \in L} \sum_{m \in M} PT_{ilm} \times X_{ijklm} \tag{1}$$

The following depicts some of the main constraints encountered in our timetabling problem.

$$\sum_{i\in I}\sum_{k\in K_j} X_{ijklm} \le 1 \qquad (j\in J,\, l\in L,\, m\in M) \tag{2}$$

Equation (2) ensures that for a particular course, only one section can be conducted in every time period.

$$P_{ij} = \left\lceil \sum_{k\in K_j}\sum_{l\in L} \frac{\left\lceil \sum_{m\in M} X_{ijklm} \Big/ H_j \right\rceil}{S_j} \right\rceil \qquad (i\in I, j\in J) \tag{3}$$

Equation (3) indicates that if teacher i teaches at least one section of course j during H_j consecutive time periods, the value of P_{ij} will be 1, meaning that teacher i teaches course j. Here, $\lceil a \rceil$ denotes the smallest integer greater than or equal to a

$$1 \le \sum_{j\in J} P_{ij} \le N_i \qquad (i\in I) \tag{4}$$

Equation (4) represents the minimum and maximum number of courses taught by each teacher. It is assumed that each teacher has to teach at least one course.

$$LT_j \le \sum_{i\in I} P_{ij} \le UT_j \qquad (j\in J) \tag{5}$$

Equation (5) limits the number of teachers who can teach for each course. The values of LT_j and UT_j are dependent on the number of sections for course j.

$$\sum_{m\in M} X_{ijklm} = Y_{ijkl} \times H_j \qquad (i\in I, j\in J,\, k\in K_j,\, l\in L) \tag{6}$$

Equation (6) shows the relationship between variables Y_{ijkl} and X_{ijklm}. If teacher i teaches course j section k for H_j time periods on day l, the value of Y_{ijkl} is equal to 1.

$$\sum_{i\in I}\sum_{k\in K_j} Y_{ijkl} \le 1 \qquad (j\in J,\, l\in L) \tag{7}$$

Equation (7) ensures that for a particular course, at most one section can be taught each day.

$$\sum_{j\in J}\sum_{k\in K_j} X_{ijklm} \le 1 \qquad (i\in I,\, l\in L,\, m\in M) \tag{8}$$

Equation (8) ensures that each teacher can only teach at most one course section in a particular time period.

$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j} X_{ijklm} \le C \qquad (l\in L,\ m\in M) \tag{9}$$

Equation (9) represents the constraint that at each time period, the number of course sections taught could not be more than the number of classrooms available.

$$\sum_{i\in I}\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl} = S_j \qquad (j\in J) \tag{10}$$

Equation (10) states that all course sections must be scheduled.

$$\sum_{i\in I}\sum_{l\in L} Y_{ijkl} = 1 \qquad (j\in J,\ k\in K_j) \tag{11}$$

Equation (11) ensures that each course section can only be taught by one teacher.

$$X_{ijklm} = 0 \qquad (i\in I, j\notin J_i,\ k\in K_j,\ l\in L) \tag{12}$$

Equation (12) ensures that teachers will not be assigned courses that they are unable to teach.

$$\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl} = L_i \qquad (i\in I) \tag{13}$$

Equation (13) calculates the number of course sections taught by each teacher.

$$V_i = \left\lceil \frac{L_i}{5} \right\rceil \qquad (i\in I) \tag{14}$$

Equation (14) calculates the upper bound on the number of course sections taught per day for each teacher.

$$\sum_{j\in J}\sum_{k\in K_j} Y_{ijkl} \le V_i \qquad (i\in I,\ l\in L) \tag{15}$$

Equation (15) ensures that the number of course sections taught by each teacher per day will not be more than the upper bound V_i. This constraint also tries to spread out evenly the total course sections taught by a particular teacher in a week.

The next three constraints deal with the requirement of consecutive time periods. We know that every course section requires a certain number of time periods. With these constraints, if course j section k taught by teacher i requires H_j consecutive time periods and the teacher is assigned to a given period of day l as the first period to be taught for the course section, then the teacher will also be assigned to the following $(H_j - 1)$ periods.

$$\sum_{t=0}^{(H_j-1)} X_{ijkl(m+t)} \ge H_j \times U_{ijklm} \qquad (i\in I, j\in J,\ k\in K_j,\ l\in L,\ m\in \{0,\ldots,|M|\text{-}H_j\}) \tag{16}$$

In order to facilitate the modeling of this requirement, the variable U_{ijklm} is introduced. Equation (16) expresses the constraint that each course section has to be scheduled in H_j time periods consecutively. If section k of course j taught by teacher i is assigned to time period m_1 of day l, then the following $(h_j - 1)$ time periods should be assigned to the same course section. In this case, the variable U_{ijklm} would be 1, meaning that the particular course section is started at time period m_1.

$$\sum_{i\in I}\sum_{l\in L}\sum_{m=0}^{(|M|-H_j)} U_{ijklm} = 1 \qquad (j\in J, k\in K_j) \tag{17}$$

Equation (17) ensures that for each course section, only one variable U_{ijklm} is equal to 1, meaning that the starting time period of a particular course section is only one time period.

$$\sum_{i\in I}\sum_{l\in L}\sum_{m\in M} X_{ijklm} = H_j \qquad (j\in J, k\in K_j) \tag{18}$$

The number of time periods allocated to each course section has to be equal to the requirement of the course, and this is expressed by equation (18).

$$U_{ijklm} = 0 \qquad (i\in I, j\in J, k\in K_j, l\in L, m\in \{(|M|-H_j+1),\ldots,(|M|-1)\}) \tag{19}$$

A particular course section could not be started in certain time periods if the remaining time periods are less than the number of time periods required. In this case, we set the relevant variables U_{ijklm} to be zero.

$$U_{ijklm} = 0 \qquad (i\in I, j\notin J_i, k\in K_j, l\in L, m\in \{0,\ldots,(|M|-H_j)\}) \tag{20}$$

Similar to equation (12), equation (20) ensures that teachers will not be assigned to certain time periods for courses that they are unable to teach.

Note that equations (3) and (14) involve nonlinear functions of the decision variables but these can always be linearized by adding some decision variables and constraints. The maximum number of decision variables and constraints in the proposed mathematical model are $(2|I||J||K||L||M| + |I||J||K||L| + |I||J|+2|I|)$ and $(3|I||J||K||L||M| + 2|I||J||K||L| + |I||J||L||M| + |I||L||M| + |I||J| + 3|J||K| + |I||L| + |J||L| + |L||M| + 4|I| + 3|J|)$, respectively.

To test the performance of the proposed model, it is implemented in ILOG OPL Studio 4.2 on a 2.6GHz Pentium IV PC with 512MB RAM that runs in Microsoft Windows XP operating system. Several data sets are generated in such a way that the data sets correspond to differing values of several parameters. Here, the parameters that were varied are the number of teachers, the number of courses, the number of sections for each course, the maximum load and the number of classrooms available.

The number of days per week is assumed to be five days and the number of time periods per day is eight periods except for problem type 5×5. Each data set consists of five randomly generated data instances. Two different types of data sets are generated and they are known as Group I and Group II data sets respectively. For Group I data sets, the number of sections for each

course is set to a fixed number, while the number of sections for each course varies in the Group II data sets. Tables 1 and 2 summarize the characteristics of each data set. For all the data sets, we also set the minimum and maximum number of teachers who can teach a particular course as shown in Table 3.

Table 1. The characteristics of Group I data sets

Data set	Number of teachers	Number of courses	Number of sections	Number of days	Number of time periods per day	Maximum load per teacher	Number of classrooms available
5×5_1	5	5	2	5	4	1	4
5×5_2	5	5	2	5	4	2	4
10×10_1	10	10	2	5	8	1	4
10×10_2	10	10	2	5	8	2	4
15×15_1	15	15	2	5	8	1	6
15×15_2	15	15	2	5	8	2	6
20×20_1	20	20	2	5	8	1	8
20×20_2	20	20	2	5	8	2	8

Table 2. The characteristics of Group II data sets

Data set	Number of teachers	Number of courses	Minimum number of sections	Maximum number of sections	Number of days	Number of time periods per day	Maximum load per teacher	Number of classrooms available
10×20_1	10	20	2	3	5	8	4	10
10×20_2	10	20	2	4	5	8	4	10
20×30_1	20	30	2	3	5	8	3	15
20×30_2	20	30	2	4	5	8	3	15
20×40_1	20	40	2	3	5	8	4	15
20×40_2	20	40	2	4	5	8	4	15
30×60_1	30	60	2	3	5	8	4	20
30×60_2	30	60	2	4	5	8	4	20

Table 3. The minimum and maximum number of teachers for each course

Number of sections	Minimum number of teachers (LT)	Maximum number of teachers (UT)
1	1	1
2	1	2
3	1	2
4	2	3

The maximum size of problems solvable within reasonable time is rather small due to the huge number of constraints and decision variables involved. Tables 4 and 5 summarize the average best known/optimal objective function values obtained and the average CPU time required to obtain the solutions for Group I and Group II data sets, respectively.

From Table 4, we observe that the average CPU time required to obtain the solution increases rapidly when the maximum load increases from one to two courses. The average objective function value of the optimal solutions is also increased when the maximum load is increased. This is because the chances for each teacher to be assigned the courses and time periods preferred will be higher and each course can be taught by more than one teacher. Similar observations can also be found from the results in Table 5.

Table 4. Computational results of Group I data sets

Data set	Average objective function value	Average CPU time (in seconds)
5×5_1	1,052	2.03
5×5_2	1,216	2.17
10×10_1	2,184	18.47
10×10_2	2,712	31.79
15×15_1	3,170	136.93
15×15_2	4,166	431.94
20×20_1	4,368	337.98
20×20_2	5,774	4,212.92

Table 5. Computational results of Group II data sets

Data set	Average objective function value	Average CPU time (in seconds)
10×20_1	7,766	1,183.11
10×20_2	7,934	1,273.15
20×30_1	10,875	9,746.36
20×30_2	13,468	36,019.99
20×40_1	-	-
20×40_2	-	-
30×60_1	-	-
30×60_2	-	-

The best known/optimal solution can be found for data sets with the number of teachers = 10 within a few minutes. For 20 teachers, we observed that the CPU time increases rapidly as shown in Table 5. The solutions can only be found within three to six hours. However, the optimal

solution for all data instances in data sets 20×40_1, 20×40_2, 30×60_1 and 30×60_2 could not be computed within the time limit of 24 hours.

These numerical results indicate that the computing time required to find an optimal solution to the problem becomes prohibitively large when the problem size increases. This also experimentally supports the theoretical results on the NP-completeness of timetabling problems [21]. Consequently, many heuristic algorithms were proposed and developed to deal with the timetabling problem in the literature as mentioned in the previous section. In the next section, we also propose a heuristic that can handle large problem sizes.

4 The Proposed Heuristic

Our problem consists of two common sub-problems in the university course timetabling problem: teacher assignment and course scheduling problems. Our proposed heuristic will solve these sub-problems iteratively, and it comprises of three main phases: (1) pre-processing, (2) construction, and (3) improvement (see Figure 1). Each of these phases is further described below.

Pre-Processing Phase
Generate I_j and LM_i
Construction Phase
Allocate teachers to courses and course sections
Allocate course sections to time periods
Improvement Phase
Reallocate teachers to courses and course sections followed by reallocating to time periods

Fig. 1. The proposed heuristic algorithm

4.1 Pre-Processing Phase

Two processes are involved in this phase. The given course preferences for each teacher i being allocated to each course j in such a way that each course j has a list of teachers who are willing to teach, are sorted in non-increasing order, which is called I_j. The given time period preferences (day l and time period m) for each teacher i are also sorted in non-increasing order, which is called LM_i. The time complexity for these processes is $O(|I|^2|J|)$ and $O(|I||L|^2|M|^2)$ respectively.

4.2 Construction Phase

In this phase, a feasible solution is constructed by accommodating as many course and time period preferences as possible. Each course should be allocated to the teacher who has the highest preference and scheduled to the time periods with the highest time period preferences as well. The

entire phase is started with the allocation of teachers to courses and course sections (teacher assignment problem).

In order to generate a feasible initial solution, a *Checking procedure* has to be conducted. The purpose of this procedure is to prevent a teacher from teaching more than the maximum load allowed (in terms of the number of courses). When the selected teacher has reached or exceeded the maximum load, the next teacher with less preference will be considered. In this construction phase, we assume that the number of selected teachers for each course depends on the minimum requirement (see Table 3).

However, it is possible that we will not be able to find any teacher from the list I_j. For this case, we can choose a teacher in list I_j who has the lowest number of courses allocated by relaxing requirement *b* in Section 2. However, an infeasible initial solution could be generated. Some teachers have to teach more than the maximum load, while other teachers might not teach any course, thereby resulting in infeasibility because of violation of the requirement *c* in Section 2.

Due to the infeasibility issue in the teacher assignment sub-problem above, we can try to improve it before continuing to the next step. We classify teachers into several groups: teachers who teach more than the maximum load allowed (Group 1), teachers who teach equal to or less than the maximum load allowed (Group 2) and teachers who are not allocated to any course (Group 3). The number of teachers in group *s* is represented as $|Group_s|$, for $s = 1$, 2 or 3. The aim is to replace teachers from Group 1 with other teachers until none of the teachers is in Group 1, and to allocate at least one course to the teachers in Group 3 as well. When we replace a teacher in Group 1, we have to consider all the courses taught by him/her and find another teacher who has the minimum load. A similar idea is used for allocating a teacher in Group 3. A course that is currently taught by a teacher who has the maximum load is chosen and would be allocated to a teacher from Group 3. These additional steps are conducted until none of the teachers is in Groups 1 and 3, or the total number of iterations reaches a predetermined maximum number of iterations.

The complexity of the above process is $O(|I|^2|J|)$. It is possible to redesign the procedure and make it more efficient, such as by ignoring the maximum and minimum load constraints of equation (4) in the *Checking procedure*. For this case, the complexity can be reduced to $O(|I||J|)$. However, the possibility of obtaining an infeasible solution might be higher. It would require some additional efforts, such as increasing the number of iterations or increasing the computational time in the improvement phase, in order to achieve better solutions. Also, for special cases such as when the number of sections for each course is set to 1, the complexity will be automatically reduced to $O(|I||J|)$.

For the initial allocation of time periods, each teacher has the time periods sorted based on his/her preferences. The number of time periods allocated to each teacher is dependent on the number of courses and course sections allocated during course assignment. The idea of the proposed algorithm is almost similar to course assignment. We allocate the time periods to teachers based on their time preferences.

Initially, we choose the first time period with the highest preference as the starting time period. The *Checking procedure* has to be invoked in order to address the feasibility issues. A similar

situation with the teacher assignment sub-problem will be faced if until the last time period in the list, the course section has not been allocated yet. For this situation, we have to relax some constraints and try to find other time periods. Due to the relaxed constraints, it turns out that some courses might be conducted more than once on the same day, or some time periods have the number of course sections taught that exceed the capacity, or some teachers do not have evenly spread out schedules. These courses, time periods and teachers are kept in Excess lists 1, 2 and 3, respectively. The number of members in an Excess List e is represented as $|EL_e|$, for e = 1, 2 or 3.

In order to deal with these infeasibility issues, the solution could be improved before continuing to the next phase. This is achieved by reallocating some courses and course sections to new time periods in such a way that all excess lists become empty. It is important to note that an infeasible initial solution might still occur. However, with the additional steps, the chances of infeasibility would be less. The worst case time complexity of this course scheduling process is $O(|I||J||K||L||M|)$. Again, this time complexity can be reduced through efficient procedures or for some special cases.

After performing the entire processes described above, the course and time objective function values are then calculated, before adding them together to obtain the total objective function value. The details of these calculations are explained next:

$$\textit{Course objective function value} = \sum_{i\in I}\sum_{j\in J} PC_{ij}\times P_{ij} + \left[\left|Group_1\right| + \left|Group_3\right|\right]\times PENALTY1 \tag{21}$$

$$\textit{Time objective function value} = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\sum_{m\in M} PT_{ilm}\times X_{ijklm} + \left[\left|EL_1\right| + \left|EL_2\right| + \left|EL_3\right|\right]\times PENALTY2 \tag{22}$$

$$\textit{Total objective function value} = \textit{course objective function value} + \textit{time objective function value} \tag{23}$$

Here, the terms PENALTY1 and PENALTY2 reflect the penalty values for the violations of the respective requirements listed in Section 2.

4.3 Improvement Phase

In this phase, after an initial solution is obtained from the previous phase, two operations are performed in order to seek further improvement. These two operations are re-allocation of teachers to courses and course sections, followed by re-scheduling of these changes into days and time periods. The initial solution is treated as a starting solution for the improvement phase.

All the operations in this phase explore the possible neighborhood to select the next movement. The first operation is started by choosing course j randomly followed by finding another teacher without violating the maximum load constraint. Two possible alternatives are considered: the new

teacher will be added to the list of teachers who teach course j or the new teacher will replace the teacher who has been allocated to course j. If the number of teachers who teach course j is still less than the maximum number of teachers allowed, UT_j, the two alternatives are chosen randomly. Otherwise, only the last alternative can be chosen.

The second operation is to schedule the changes in teacher assignment. The new teacher is allocated to the previous day and time periods scheduled for the previous teacher, if possible. We have to make sure that at these time periods, the new teacher has not been scheduled yet and he has an evenly spread out schedule as well. Otherwise, we have to find a new set of days and time periods by considering some constraints, such as the length of course should not exceed the last time period on a day.

If these two operations can give a better objective function value, we will update the starting solution. Otherwise, we return to the previous starting solution. These two operations are performed until the total number of iterations reaches a predetermined maximum number of iterations. Finally, the solution of the problem is the best solution obtained so far. In general, the time complexity of the neighborhood exploration is $O(|I||J||K||L||M|)$.

Though the entire heuristic described here is a simple improvement heuristic, it is worth noting that the procedures in the improvement phase allow for various types of modifications, such as incorporating metaheuristic procedures to the improvement phase. Such suitable hybridization of the heuristic methods may exhibit significantly better performance in terms of solution quality and the time to obtain the solutions, as noted by Purchinger and Raidl [22].

5 Computational Results

To evaluate the performance of the improvement heuristic, we compare the solutions obtained by it with the solutions obtained by solving the integer programming model. The data sets mentioned in Section 3 were used again in our computational experiments on the heuristic. The entire heuristic was coded in C++ and tested on the Intel Pentium IV 2.6 GHz PC with 512 MB RAM under the Microsoft Windows XP Operating System. The parameters of the proposed algorithms are chosen experimentally to ensure a compromise between the computational time and the solution quality. The values of the parameters used in the computational study are summarized as follows: the number of iterations in Construction Phase (teacher assignment problem) = $40|I|$, the number of iterations in Construction Phase (course scheduling problem) = $20|J|$, the number of iterations in Improvement Phase = $|I||J||L||M|$, and PENALTY1, PENALTY2 = 1,000.

Table 6 and 7 summarize the results obtained from the proposed heuristic for Group I and Group II data sets, respectively. A comparison of the objective function values and computational times (in seconds) obtained by the heuristic and the integer programming model was also presented. The percentage of solution deviation from best known/optimal objective function value is defined by *Pct* = (*best known/optimal objective function value – objective function value obtained by* the *heuristic*) / (*best known/optimal objective function value*) × 100.

Table 6. The comparison of the heuristic results and the optimal solutions (Group I data sets)

Data set	Solution obtained by OPL		Heuristic		Average *Pct* of objective function value
	Average objective function value	Average CPU time (in seconds)	Average objective function value	Average CPU time (in seconds)	
5×5_1	1,052	2.03	924	0.01	12.17
5×5_2	1,216	2.17	988	0.01	18.75
10×10_1	2,184	18.47	1,896	0.03	13.19
10×10_2	2,712	31.79	2,106	0.13	22.35
15×15_1	3,170	136.93	2,818	0.11	11.10
15×15_2	4,166	431.94	3,050	0.43	26.79
20×20_1	4,368	337.98	3,956	0.27	9.43
20×20_2	5,774	4,212.92	4,100	1.14	28.99

Table 7. The comparison of the heuristic results and the optimal solutions (Group II data sets)

Data set	Solution obtained by OPL		Heuristic		Average *Pct* of objective function value
	Average objective function value	Average CPU time (in seconds)	Average objective function value	Average CPU time (in seconds)	
10×20_1	7,766	1,183.11	6,228	0.36	19.80
10×20_2	7,934	1,273.15	6,394	0.29	19.41
20×30_1	10,875	9,746.36	8,186	2.83	24.73
20×30_2	13,468	36,019.99	10,246	4.33	23.92
20×40_1	-	-	10,340	5.76	-
20×40_2	-	-	12,880	11.90	-
30×60_1	-	-	16,064	69.13	-
30×60_2	-	-	18,962	78.58	-

We observe that the proposed heuristic can yield good solutions with the percentage of solution deviation from the best known/optimal solutions being less than 29%. We also notice that the proposed heuristic is able to find the solutions within a reasonable amount of computational time. Some problems that were unsolved by the ILOG OPL Studio software can also be easily solved by the proposed heuristic, as can be seen in data sets such as that of 20×40_1, 20×40_2, 30×60_1, and 30×60_2.

Table 8. Load distribution of teachers

Data Set	Maximum load	Average percentage of teachers having the following load				Load variance
		1	2	3	4	
5×5_1	1	100%	0%	0%	0%	0.00
5×5_2	2	72%	28%	0%	0%	0.16
10×10_1	1	100%	0%	0%	0%	0.00
10×10_2	2	41.33%	25.33%	0%	0%	0.23
15×15_1	1	100%	0%	0%	0%	0.00
15×15_2	2	64%	36%	0%	0%	0.22
20×20_1	1	100%	0%	0%	0%	0.00
20×20_2	2	74%	26%	0%	0%	0.18
10×20_1	4	8%	30%	34%	28%	0.84
10×20_2	4	6%	20%	30%	44%	0.85
20×30_1	3	24%	46%	30%	0%	0.53
20×30_2	3	20%	39%	41%	0%	0.56
20×40_1	4	13%	27%	35%	25%	0.95
20×40_2	4	14%	15%	28%	43%	1.13
30×60_1	4	9.33%	30.67%	34%	26%	0.88
30×60_2	4	6.67%	21.33%	35.33%	36.67%	0.84

Another observation of interest is on the load distribution of the teachers with respect to the maximum load that we specify (see Table 8). For Group I data sets, we notice that the proposed heuristic distributes the load evenly to teachers. The load variances obtained are reasonably small. However, for Group II data sets, the load variances are found to be large with only a small percentage of teachers having very small load.

6 Conclusions

We have proposed a new mathematical programming model for a timetabling problem that combines two sub-problems, teacher assignment and course scheduling, simultaneously. This model allows the possibility that courses could be taught by more than one teacher and each course might consist of more than one sections.

An improvement heuristic was proposed for solving the timetabling problem that cannot be easily solved by commercial software. The proposed heuristic solves these sub-problems iteratively. The computational results show that the proposed heuristic yields good solutions within reasonable computational time. The results obtained from the proposed heuristic were also compared against solutions of an integer programming approach.

As possible future research directions, we can look into ways of improving the proposed heuristic to obtain solutions of better quality even when the problem size is very large. This would include the hybridization of the proposed heuristic with other types of heuristics as mentioned previously. Since each university has different characteristics and requirements, the mathematical programming model and the proposed heuristic can also be extended to adapt to these distinct characteristics and requirements.

References

1. Wren, A.: Scheduling, timetabling and rostering – A special relationship? In E. Burke and P. Ross (ed), Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science **1153,** Springer-Verlag, New York (1996) 46-75
2. Carter, M. W., Laporte, G.: Recent developments in practical course timetabling. In E. Burke and M. Carter (ed), Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science **1408,** Springer-Verlag, New York (1998) 3-19
3. Andrew, G. M., Collins, R.: Matching faculty to courses. *College and University* **46** (1971) 83-89
4. Breslaw, J. A.: A linear programming solution to the faculty assignment problem. *Socio-Economic Planning Sciences* **10** (1976) 227-230
5. Harwood, G. B., Lawless, R. W.: Optimizing faculty teaching schedules. *Decision Science* **6** (1975) 513-524
6. Schniederjans, M. J., Kim, G. C.: A goal programming model to optimize departmental preference in course assignments. *Computers and Operations Research* **14** (1987) 87-96
7. Tillett, P. I.: An operations research approach to the assignment of teachers to courses. *Socio-Economic Planning Sciences* **9** (1975) 101-104
8. Wang, Y. Z.: An application of genetic algorithm methods for teacher assignment problems. *Expert Systems with Applications* **22** (2002) 295-302
9. Yu, E., Sung, K. S.: A genetic algorithm for a university weekly courses timetabling problem. *International Transactions in Operational Research* **9** (2002) 703-717
10. Abramson, D.: Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science* **37** (1991) 98-113
11. Abramson, D., Krishnamoorthy, M., Dang, H.: Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pacific Journal of Operational Research* **16** (1999) 1-22
12. Hertz, A.: Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics* **35** (1992) 255-270
13. Badri, M. A.: A two-stage multiobjective scheduling model for [faculty-course-time] assignments. *European Journal of Operational Research* **94** (1996) 16-28
14. Daskalaki, S., Birbas, T., Housos, E.: An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research* **153** (2004) 117-135

15. Gunawan, A.., Ng, K.M., Poh, K.L.: A Mathematical Programming Model for a Timetabling Problem. *Proceedings of the International Conference on Scientific Computing*, *June 2006*, *Nevada, USA* (2006)
16. Costa, D.: A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research* **76** (1994) 98-110
17. Eiselt, H.A., Laporte, G.: Combinatorial optimization problems with soft and hard requirements. The Journal of the Operational Research Society **38** (1987) 785-795
18. Aubin, J., Ferland, J.A.: A large scale timetabling problem. *Computers and Operations Research* **16** (1989) 67-77
19. Loo, E.H., Goh, T.N., Ong, H.L.: A heuristic approach to scheduling university timetables. *Computers and Education* **10** (1985) 379-388
20. Wright, M.: School timetabling using heuristic search. *The Journal of the Operational Research Society* **47** (1996) 347-357
21. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. *SIAM Journal of Computing* **5** (1976) 691-703
22. Puchinger, J. Raidl, G.R.: Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In J. Mira and J.R. Alvarez (ed), IWINAC 2005, Lecture Notes in Computer Science **3562,** Springer-Verlag, New York (2005) 41-53